Agiletek Coding Exam

This exam is to be carried out by potential new starters. It is designed to give us an idea of a candidates problem solving skills across a range of problems that we frequently encounter. Try to spend a couple of hours on this – but don't worry if you can't get it all finished.

The specification

We have a CSV file (***input.csv)*** containing CPS (Cable Protection System) data that needs to be parsed and then the data posted to an API which handles saving the data into a database. The ***input.csv*** file only contains a few rows and columns – but a real example of this CSV could contain 50-100 rows and potentially thousands of columns.

We want you to write a python function that will parse this CSV, format the data as required by the API and then POST the data as JSON to **`https://api.agiletek.co.uk/cps`**.

After all the rows are processed the function should output a ***successful.csv*** and a ***failed.csv***. The ***successful.csv*** file should just contain the data that was sent to the API (one row per API call). The ***failed.csv*** should contain data that was not sent to the API (I.e. because the row was missing data that the API requires)

Important points

- This API doesn't actually exist – you will have to figure out how to build a python function and write tests that send data to a non-existent API.
- The API only accepts a single row per request
- The API only accepts requests in JSON format
- The API response will be in JSON format (see **`example_api_response.json`**)
- **All fields are required**. Any row with missing data should not be posted to the API and should instead be written to `failed.csv`
- The API only accepts fields of the correct data type. If the field data is the wrong type (based on **`example_options.json`**) then the row should be written to **failed.csv**
- There is no rate limit on the API
- See "***example_options.json***" for details of the fields that the API requires
    - o Note that there are some fields that the API requires that are not in the CSV. These fields will need to be calculated by you prior to sending them to the API. These fields are below:

        - **cable_mg_volume** = cable_mg_thickness * cable_mg_width * cable_mg_length
        - **project_code** - All project codes should be prepended with `TEK-` before sending the data to the API

- See "***example_api_response.json"*** to see an example response from the API

- The `**input.csv**` only contains a few columns and rows. However, in a live system the number of columns and rows could be far greater. Your code will need to be written in a way that will allow us to add extra columns and rows to the `**input.csv**` without having to add too much extra complexity to the code. Remember that network connections are slow, so we don't necessarily want to have to sit and wait while 1000 network connections are made consecutively

What we want to see

- A script that will accept a filepath to `**input.csv**`. This should then parse the file, clean and process the code, and then submit it to the API. We should then get a `**successful.csv**` and ` **failed.csv**` as an output.
    - `successful.csv ` should contain rows that were sent to the API
    - `failed.csv` should contain rows that were not sent to the API (because the data was incomplete or incorrect)

- We want to see examples of tests
    - These tests should be runnable (it doesn't matter if they fail – but we want to see evidence that they exist)
    - At the minimum there should be an integration test that proves that the API received the expected data and returned a response – however, there are other areas where extra unit tests could be beneficial

- The code should be OS agnostic – We should be able to run this script (And the tests) regardless of the operating system

- We'd like to see a small readme with instructions for how to run your script and how to run the tests

Nice to haves

- If you have time it would be great to have this as an installable package

- If you have time it would be great to have a CLI to run this script from. The CLI should be in the format:
    - `python <script.py> --filepath=<path/to/input.csv>`